Adaptive Backdoor Attacks with Reasonable Constraints on Graph Neural Networks

Xuewen Dong, *Member, IEEE,* Jiachen Li, Shujun Li, *Senior Member, IEEE,* Zhichao You, Qiang Qu, Yaroslav Kholodov, and Yulong Shen, *Member, IEEE*

***Abstract*—Recent studies show that graph neural networks (GNNs) are vulnerable to backdoor attacks. Existing backdoor attacks against GNNs use fixed-pattern triggers and lack reasonable trigger constraints, overlooking individual graph characteristics and rendering insufficient evasiveness. To tackle the above issues, we propose ABARC, the first Adaptive Backdoor Attack with Reasonable Constraints, applying to both graph-level and node-level tasks in GNNs. For graph-level tasks, we propose a subgraph backdoor attack independent of the graph's topology. It dynamically selects trigger nodes for each target graph and modifies node features with constraints based on graph similarity, feature range, and feature type. For node-level tasks, our attack begins with an analysis of node features, followed by selecting and modifying trigger features, which are then constrained by node similarity, feature range, and feature type. Furthermore, an adaptive edge-pruning mechanism is designed to reduce the impact of neighbors on target nodes, ensuring a high attack success rate (ASR). Experimental results show that even with reasonable constraints for attack evasiveness, our attack achieves a high ASR while incurring a marginal clean accuracy drop (CAD). When combined with the state-of-the-art defense randomized smoothing (RS) method, our attack maintains an ASR over 94%, surpassing existing attacks by more than 7%.**

***Index Terms*—Graph neural networks, backdoor attacks, trigger constraint, backdoor evasiveness**

I. Introduction

GRAPH-STRUCTURED data play a crucial role in the real world by effectively modeling and analyzing intricate relationships and interconnectedness between entities [1]. For instance, a chemical molecular structure can be abstracted as graph data [2], where atoms can be regarded as nodes, and the chemical relationship between atoms can be represented by edges. Graph neural networks (GNNs) [3], [4] are proposed to learn representations encompassing node features, topology, and neighbor relationships within graph-structured data [5]. A major task of GNNs is graph-level tasks, such as graph classification, which takes a graph as an input and outputs a label for the graph. Graph classification is a basic graph analytics tool and has many applications such as malware detection [6], and healthcare [7]. Besides, some existing studies on GNNs focus on node-level tasks, such as node classification, which aims to predict a label for each node in a graph. Node classification has also been applied in many scenarios, such as fraud detection [8], and user preference judgment [9], [10], [11].

Due to the increasing popularity of GNNs, their security has become a major concern [12], [13], [14], [15]. One of the most concerning types of security threats is backdoor attacks [16], [17], which can result in disastrous consequences, such as misdiagnosed health conditions and privacy leakage. A backdoor attack is an adversarial attack that involves inserting a backdoor into a model during the training phase. A backdoor is a specific pattern (trigger) of the inputs that, when presented to the model, will cause it to produce a specific output or prediction for the adversary's malicious benefit. The goal of a backdoor attack is to manipulate the model's behavior in a targeted way without affecting its overall performance on non-targeted legitimate inputs. Existing attacks focus on node-level and graph-level tasks of GNNs [18], [19], [20], [21].

While existing backdoor attacks against GNNs have achieved a specific ASR, they face the problem of insufficient evasiveness due to their limitations on using fixed pattern triggers and the lack of reasonable constraints on triggers, which make detection easier. For graph-level tasks, most existing attacks rely on fixed patterns for all target objects and overlook the distinctive characteristics of individual graph samples, such as topological structures, node features, and edge relationships [18], [19]. Additionally, existing adaptive attacks lack reasonable trigger constraints, increasing their detectability [20]. Regarding node-level tasks, some existing attacks overlook that GNNs incorporate the attributes of the node and its neighboring nodes as high-dimensional features, therefore neglecting the impact of neighboring nodes on the target node and resulting in a low ASR [19]. Other attacks that consider the above issues weaken the influence of the target node's neighbor nodes by adding new nodes and edges to the target node, but they can be defended by RS. Besides, existing

This work was supported in part by the National Key R&D Program of China (No. 2023YFB3107500), National Natural Science Foundation of China (No. 62220106004, 62232013), the Technology Innovation Leading Program of Shaanxi (No. 2022KXJ-093, 2023KXJ-033), the Innovation Capability Support Program of Shaanxi (No. 2023-CX-TD-02), the Fundamental Research Funds for the Central Universities (No. ZDRC2202), and Basic and Applied Basic Research Foundation of Guangdong Province (No. 2023TQ07A264). (Corresponding author: Xuewen Dong.)

Xuewen Dong, Jiachen Li, and Zhichao You are with the School of Computer Science and Technology, Xidian University, the Engineering Research Center of Blockchain Technology Application and Evaluation, Ministry of Education, and also with the Shaanxi Key Laboratory of Blockchain and Secure Computing, Xi'an 710071, China (e-mail: xwdong@xidian.edu.cn; jiachenl@stu.xidian.edu.cn; zcyou@stu.xidian.edu.cn).

Shujun Li is with the School of Computing & Institute of Cyber Security for Society (iCSS), University of Kent, Canterbury, CT2 7NP, U.K. (e-mail: S.J.Li@kent.ac.uk).

Qiang Qu is with Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences (e-mail: qiang@siat.ac.cn).

Yaroslav Kholodov is with the Intelligent Transportation Systems Lab, Innopolis University, Innopolis, Russia (e-mail: ya.kholodov@innopolis.ru).

Yulong Shen is with the School of Computer Science and Technology, Xidian University, and also with the Shaanxi Key Laboratory of Network and System Security, Xi'an 710071, China (e-mail: ylshen@mail.xidian.edu.cn).

attacks for node-level tasks also lack reasonable constraints, rendering insufficient evasiveness [20], [21].

There are multiple technical **challenges** on addressing the above-mentioned problems of backdoor attacks: (1) *Intricate trigger design*. Considering that setting fixed pattern triggers for all graphs can be easily detected, we set a unique trigger for each graph. The design of a unique trigger scheme is intricately related to factors such as node selection, topology, and feature distribution. When confronted with multiple factors, devising unique triggers that align with the characteristics of each factor combination becomes a challenging task. (2) *Reasonable constraint construction*. Adding reasonable constraints yields a negative effect on the success rate of attacks. Ensuring attack evasiveness hinges on precise constraint formulation and imposition on triggers, highlighting the crucial need to strike a delicate balance between achieving a high attack success rate and maintaining undetectability. (3) *Specific tasks adjustment*. The effective backdoor attacks in graph-level and node-level tasks bring the problem of task suitability. Graph-level tasks of GNNs focus more on the information of the entire graph. The role of the graph's topology in graph analysis is paramount. More specifically, modifying the graph's topology in existing GNN backdoor trigger schemes leads to ease of detection. For node-level tasks, GNNs incorporate the attributes of the node and its neighboring nodes as high-dimensional features. Effectively reducing the influence of neighboring nodes on the target node while ensuring the undetectability of the trigger-embedded target node presents a significant challenge.

Our main contributions are:

- **Adaptive triggers**. For graph-level tasks, we select nodes randomly without considering the importance of nodes as subgraph triggers according to the node size of graph samples. We define a node feature modification formula as an adaptive pattern. For node-level tasks, we select node features according to the importance of the features and feature dimension. We also define a node feature modification formula as an adaptive pattern.
- **Reasonable constraints**. We impose constraints on the trigger for both graph-level and node-level tasks in GNNs by evaluating its impact on the similarity between the target object with the trigger and the target object without the trigger. Besides, we consider the value range of each node feature and the value type of each node feature to ensure the trigger node features would not be outliers or obviously wrong values. Furthermore, we give two examples to demonstrate our trigger-generation methods for both graph-level and node-level tasks in GNNs.
- **Task adjustment mechanism**. We propose specific mechanisms for both graph-level and node-level tasks to ensure the effectiveness and the evasiveness of our attack. For graph-level tasks, we propose a topology-free subgraph trigger generation method, which uses an entirely randomized node selection mechanism tailored to the specific node size of the graph samples. For node-level tasks, we have devised an adaptive edge-pruning mechanism aimed at achieving a notably high ASR.
- **Evaluation**. We conducted experiments across multiple models and datasets, demonstrating the generalizability and effectiveness of our attacks. To evaluate the robustness of our attacks, we compared them against two state-of-the-art (SOTA) defense methods. When combined with the SOTA defense randomized smoothing (RS) method, our attacks maintain an ASR over 94%, surpassing existing attacks by more than 7%. Furthermore, our attacks also remain undetectable even when another defense method – neural cleanse (NC) – is applied.
- **GNN vulnerabilities exposure**. In the design of adaptive triggers, reasonable constraints, and task adjustment mechanisms, we have revealed the security flaws of GNNs. We enhance the understanding of how backdoor attacks can be more adaptive and resilient against existing defenses. Our introduction of adaptive triggers and reasonable constraints specifically targets the security weaknesses in GNNs. The proposed task adjustment mechanisms ensure that our attacks are not only effective but also evasive, posing a substantial challenge to current defense methods. Through rigorous evaluation, we highlight the limitations of existing defense mechanisms, providing critical insights into the security vulnerabilities of GNNs.

The remainder of this paper is organized as follows. We introduce research related to backdoor attacks against deep neural networks (DNNs) and GNNs in Section 2, while the background of GNNs and backdoor attacks is given in 3. In Section 4, we present our adaptive graph-level backdoor attack. Next, we describe our adaptive node-level backdoor attack in Section 5. The performance of ABARC is evaluated in Section 6, and we conclude our works in Section 7.

II. Related Work

A. Advanced GNNs and Graph Learning

Graph Neural Networks (GNNs) have seen significant advancements in recent years, driven by the need to generalize deep learning methods to graph-structured data. This need has led to the development of various models and approaches to enhance the performance and applicability of GNNs across different domains [22].

Early GNN models faced challenges when applied to heterophilous graphs, where nodes with different labels or features are more likely to connect. This limitation prompted the exploration of aggregations beyond the one-hop neighborhood. To address the above limitation, researchers developed models that implement multiscale extraction via constructing Haar-type graph framelets. These framelets exhibit desirable properties such as permutation equivariance, efficiency, and sparsity, making them suitable for deep learning tasks on graphs. For instance, the Permutation Equivariant Graph Framelet Augmented Network (PEGFAN) [23] leverages these framelets to achieve SOTA performance on several heterophilous graph datasets, demonstrating its efficacy in handling complex graph structures.

Recommender systems have also benefited from the advancements in GNNs, particularly through integrating contrastive self-supervised learning (SSL) methods. GNN-based SSL approaches have outperformed traditional supervised

TABLE I
RELATED WORKS SUMMARY (GRAPH-LEVEL | NODE-LEVEL).

Attack	Tasks	Trigger	Adaptive trigger	Topology-free	Constraints reasonability
BKD [18]	✓| ✗	subgraph | -	✗| -	✗| -	✗| -
EXP [19]	✓| ✓	subgraph | node features	✗| ✗	✗| ✓	✗| ✗
GTA [20]	✓| ✓	subgraph | subgraph	✓| ✓	✗| ✗	✗| ✗
UGBA [21]	✗| ✓	- | subgraph	- | ✓	- | ✗	- | ✗
ABARC (Ours)	✓| ✓	subgraph | node features	✓| ✓	✓| ✗	✓| ✓

learning paradigms in graph-based recommendation tasks. However, these methods often require extensive negative examples and complex data augmentations. The Bootstrapped Graph Representation Learning with Local and Global Regularization (BLoG) [24] model addresses these challenges by constructing positive/negative pairs based on aggregated node features from alternate views of the user-item graph. BLoG employs an online and target encoder, introducing local and global regularization to facilitate information interaction. This innovative approach has shown superior recommendation accuracy on benchmark datasets compared to existing baselines.

Graph Convolutional Networks (GCNs) are a cornerstone of GNNs, recognized for their ability to learn node representations effectively. Various extensions to GCNs have been proposed to improve their performance, scalability, and applicability. One notable advancement is the introduction of random features to accelerate the training phase in large-scale problems. The Graph Convolutional Networks with Random Weights (GCN-RW) [25] model revises the convolutional layer with random filters and adjusts the learning objective using a regularized least squares loss. Theoretical analyses of GCN-RW's approximation upper bound, structure complexity, stability, and generalization have been provided, demonstrating its effectiveness and efficiency. Experimental results indicate that GCN-RW can achieve comparable or better accuracies with reduced training time compared to SOTA approaches.

B. Backdoor attacks against GNNs

Some researchers have studied backdoor attacks on GNNs, as shown in Table I. For graph classification tasks, existing backdoor attacks used subgraphs as triggers. Zhang et al. [18] utilized subgraphs with fixed patterns as triggers, which we refer to as BKD. However, they did not consider the influence of node features. Xu et al. [19] also used subgraphs with a fixed pattern as triggers, which we refer to as EXP. They used GNNExplainer [26] to explain GNN-based model predictions by identifying a small, influential subgraph and node features within the input graph, which they used as a trigger. However, they did not consider the importance of node features, either. Xi et al. [20] proposed graph trojaning attack (GTA), which comprehensively considers the topology structure and node features of subgraph triggers and dynamically generates triggers for each attack sample, which can achieve a higher ASR. However, they did not consider reasonable constraints of the modified node features.

For node classification tasks, Xu et al. [19] used GraphLIME [27], a technique that explains decisions in graph models, to analyze and choose important features of a specific node, modifying them to act as a trigger. This method did not consider the message transmission process of GNNs and lacked reasonable constraints on the features of the trigger node. Both GTA proposed by Xi et al. [20], and unnoticeable graph backdoor attack (UGBA) proposed by Dai et al. [21] use subgraphs as triggers. However, they still lacked reasonable constraints on the features of trigger nodes. In addition, these two methods would involve adding multiple edges to attack a specific node, and the changes would be significant. Furthermore, Zhang et al. [28] proposed a graph contrastive backdoor attack (GCBA) for graph contrastive learning, which targets self-supervised learning of a large amount of unlabeled data. It is different from the supervised learning focused on in this paper.

III. PRELIMINARY

A. Graph Neural Networks

Graph Neural Networks (GNNs) have been developed to process non-Euclidean spatial data, such as graphs. For a graph $G = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{X})$, $\boldsymbol{V}$, $\boldsymbol{E}$, $\boldsymbol{X}$ represent nodes, edges and node features, respectively. The objective of GNNs on G is to learn an embedding representation vector $\mathbf{h}_G$ for the entire graph or $\mathbf{h}_v$ for each node $v \in \boldsymbol{V}$. GNNs can effectively capture the complex relationships between nodes and provide useful insights for various applications.

For node-level tasks, GNNs typically adopt a neighborhood aggregation approach to update node representations [29], [4], [30]. The graph convolution operation of GNNs is defined as:

$$\boldsymbol{h}_v^k = \sigma\left(\boldsymbol{h}_v^{k-1}, \mathrm{AGG}\left(\left\{\boldsymbol{h}_u^{k-1}\right\}\right); u \in \mathcal{N}_v, v \in \boldsymbol{V}\right), \quad (1)$$

where $\boldsymbol{h}_v^k$ is the representation of node v in the k-th iteration, σ is an activation function, $\mathcal{N}_v$ means the set of neighbors of node v, and AGG$(\cdot)$ is the aggregation function that could vary for different GNNs.

For graph-level tasks, GNNs need to capture the global information of the graph data, including the structural and feature information of each node. A typical approach is to use a readout function [31] to aggregate the representations of all nodes in the graph and output the global representation of the graph:

$$\boldsymbol{h}_G = \mathrm{Readout}\left(\boldsymbol{h}_v; v \in \boldsymbol{V}\right). \quad (2)$$

The readout function could be a simple permutation invariant function, such as summation, or a more sophisticated *graph-level* pooling function.

B. Backdoor Attacks

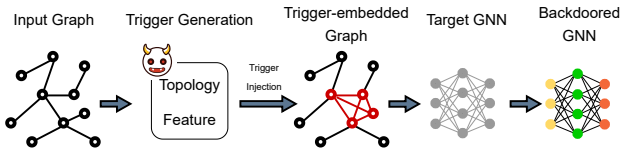


Fig. 1. The process of the backdoor attacks on GNNs.

The main objective of a backdoor attack is to implant a backdoor into a model. The adversary usually embeds triggers into some training samples. Then, the adversary uses the trigger-embedded samples to train the model during the training phase, resulting in a model that contains a backdoor. When the backdoored model is applied to non-trigger samples, it behaves normally. However, it would misclassify malicious samples containing the trigger pattern as the target class desired by the adversary [32], [33], [34]. The process of the backdoor attacks on GNNs is shown in Figure 1. Assuming that the GNN model is θ, the backdoored GNN model is θ_t and the adversary's target class is y_t. For a given non-trigger graph G, if we define such trigger-embedded graph as G_{g_t}, the adversary's objective could be defined as:

$$\begin{cases} \theta_t\left(G_{g_t}\right) = y_t, \\ \theta_t\left(G\right) = \theta\left(G\right). \end{cases} \tag{3}$$

C. Threat Model

In this paper, we consider a common black-box attack scenario.

Adversary's knowledge. We assume the adversary has access to a dataset sampled from the training data of the target model. However, the adversary does not know the architecture of the target GNN model. Such an adversary represents a more realistic scenario where black-boxed AI models can be attacked.

Adversary's capability. The adversary has the ability to modify the samples it can access and inject the modified samples into the training process of the target GNN model.

Adversary's goal. In this paper, the goal of the adversary is referring to Eq. (3). Additionally, the adversary aims to make the attack as stealthy as possible to avoid detection.

IV. Adaptive Graph-level Backdoor Attack

GNNs are instrumental in graph-level tasks like graph generation and classification. Within medical science, GNNs find frequent application in tasks involving the classification of intricate graphs, facilitating predictions related to the attributes of molecules. Notably, these networks contribute to discovering potential drugs [35], ascertaining the enzymatic nature of proteins [36], and addressing analogous inquiries. In this specific scenario, a noteworthy concern emerges in the form of potential backdoor attacks. The primary objective of these malicious actors is to exploit vulnerabilities within GNNs, surreptitiously inserting a backdoor. This surreptitious manipulation subsequently engenders the misclassification of virus or protein attributes by GNNs, thereby precipitating substantial and consequential ramifications for medical outcomes.

A. Attack Analysis

The effectiveness of the backdoor attack depends on the recognition of the trigger and the difference in the vector representation of the graph samples with and without embedded triggers. When the trigger has high recognition and when the vector representation of the graph samples with and without embedded triggers is significantly different, using the graph samples with embedded triggers to train the model can enable it to learn better the characteristics of the graph samples with embedded triggers and classify the graph samples with embedded triggers into one category, which is the target category of the adversary. Assume that a trigger is added to G_i, and its features are modified to $\boldsymbol{X}^j_{i,g_t}$, δ is the successful attack threshold. The difference in the vector representation of the graph sample embedded with the trigger and its corresponding non-embedded trigger at the output layer is:

$$\Delta_{\boldsymbol{h}_{G_i}} = \left|\left|\boldsymbol{h}_{G_{i,g_t}} - \boldsymbol{h}_{G_i}\right|\right|, \tag{4}$$

when $\Delta_{\boldsymbol{h}_{G_i}} > \delta$, the backdoor can be successfully embedded into the model, and the model can misclassify the graph sample embedded with the trigger as the adversary's target category. On the one hand, when the GNN model updates the vector representation of each node in the input layer and the hidden layer, it needs to aggregate the vector representation of the node and its neighbor nodes in the previous layer. This operation will weaken the characteristics of the node features themselves. On the other hand, GNN graph-level tasks usually use the Readout$(\cdot)$ function to process the vector representations of all nodes to obtain the final vector representation of the entire graph, which will also weaken the characteristics of the node features themselves. Therefore, in order to ensure that $\Delta_{\boldsymbol{h}_{G_i}} > \delta$, three aspects can be considered: 1) Select a small number of nodes as subgraph triggers and modify their topological structure and node features at the same time. 1) Make special modifications to the topological structure of the selected subgraph trigger; for example, modify it to a fully connected subgraph. When the model updates the feature representation of each trigger node, the influence of non-trigger nodes can be weakened because the node has multiple neighboring trigger nodes; 2) Select a small number of nodes as subgraph triggers, do not modify their topological structure, and only make a large offset to their original features; 3) Consider selecting more nodes as subgraph triggers, and also do not modify their topological structure, and make a relatively small offset to their original features.

The evasiveness of the backdoor attack depends on whether the attack significantly changes the feature distribution of the overall graph. If the feature distributions of graph samples without embedded triggers and with embedded triggers are

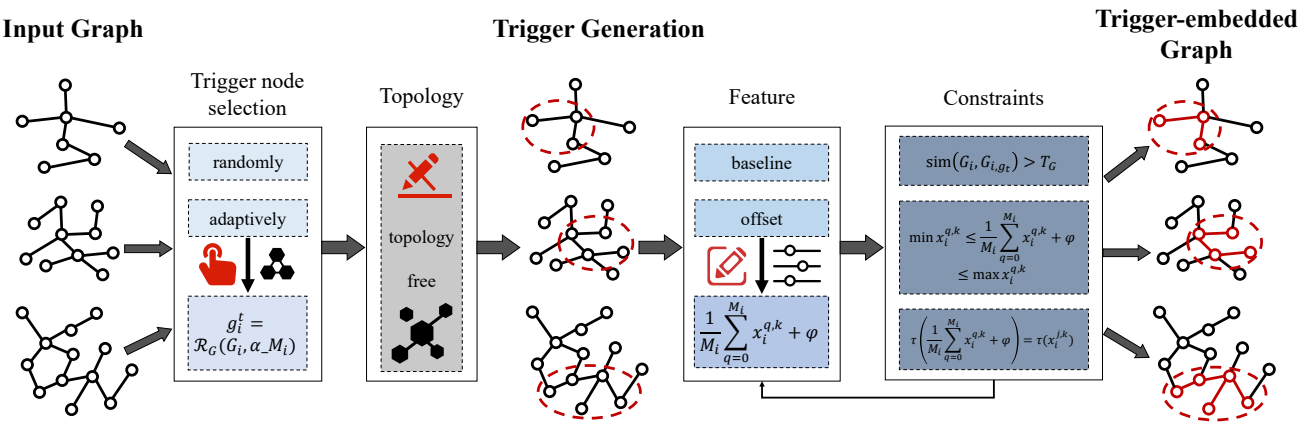


Fig. 2. The framework of our adaptive graph-level trigger generation method.

similar, the attack is considered to be covert, so cosine similarity can be used to evaluate:

$$\mathrm{sim}(G_i, G_{i,g_t}) = \frac{\langle \boldsymbol{X}_i, \boldsymbol{X}_{i,g_t} \rangle}{\left|\left|\boldsymbol{X}_i\right|\right| \cdot \left|\left|\boldsymbol{X}_{i,g_t}\right|\right|}, \tag{5}$$

$\boldsymbol{X}_i$ and $\boldsymbol{X}_{i,g_t}$ are the original features of graph samples without and with embedded triggers, respectively.

B. Attack Overview

Given a dataset $\boldsymbol{D} = \{(G_1, y_1), (G_2, y_2), \cdots, (G_N, y_N)\}$, where G_i and y_i respectively represent the i-th sample and its true label, N is the number of the samples. For a graph sample $G_i = (\boldsymbol{V}_i, \boldsymbol{E}_i, \boldsymbol{X}_i)$, $\boldsymbol{V}_i = \{v_i^1, v_i^2, \cdots, v_i^{M_i}\}$, where M_i is the node number of G_i, $\boldsymbol{E}_i \subseteq \boldsymbol{V}_i \times \boldsymbol{V}_i$ is the set of edges of G_i, and $\boldsymbol{X}_i = \{\boldsymbol{x}_i^1, \boldsymbol{x}_i^2, \cdots, \boldsymbol{x}_i^{M_i}\}$ is the set of node features, $\boldsymbol{x}_i^j = \{x_i^{j,1}, x_i^{j,2}, \cdots, x_i^{j,d}\}$ is the node feature vector of v_i^j, where d is the number of different node features. $\boldsymbol{A}_i \in \mathbb{R}^{M_i \times M_i}$ is the adjacency matrix of the graph G_i, where $\boldsymbol{A}_i^{j,k} = 1$ if nodes v_i^j and v_i^k are connected; otherwise $\boldsymbol{A}_i^{j,k} = 0$.

For a graph sample G_i, we use subgraph $g_i^t = (\boldsymbol{V}_i^t, \boldsymbol{E}_i^t, \boldsymbol{X}_i^t)$ as its trigger, where $\boldsymbol{V}_i^t$ is the set of the trigger nodes. $\boldsymbol{E}_i^t$ is the set of edges and $\boldsymbol{A}_i^t$ is the adjacency matrix of the subgraph. $\boldsymbol{X}_i^t$ is the set of node features of the trigger nodes. Assuming that the adversary could access a subset $\boldsymbol{D}_t \subseteq \boldsymbol{D}$, and the number of the samples of $\boldsymbol{D}_t$ is N_t. In order to get a backdoored model θ_t, the training process of the target model θ can be defined as:

$$\begin{aligned} \theta_t = \underset{\theta}{\mathrm{argmin}} \Bigg(& \sum_{(G_i, y_i) \in \boldsymbol{D}_c} \ell\left(\theta\left(G_i\right), y_i\right) \\ & + \sum_{(G_i, y_i) \in \boldsymbol{D}_t} \ell\left(\theta\left(\mathcal{A}_G\left(G_i\right)\right), y_t\right) \Bigg), \end{aligned} \tag{6}$$

where $\boldsymbol{D}_c = \boldsymbol{D} - \boldsymbol{D}_t$ and $\mathcal{A}_G(\cdot)$ is our subgraph trigger generation method.

Unlike fixed-pattern trigger backdoor attacks, we aim to design a more general trigger generation method. In order to ensure a high ASR and undetectability of backdoor attacks on GNNs, the following **challenges** exist: (1) *intricate trigger design*, (2) *reasonable constraint construction*, and (3) *difficulty in topology-free trigger*.

To address the above challenges, we proposed three mechanisms as follows. (1) We designed an entirely random dynamic and topology-free node selection mechanism according to the node size of the graph samples. (2) We found that for some graph data, we could only modify the features of the trigger nodes to backdoor GNNs. We provide a feature-based trigger generation method. (3) We use cosine similarity to evaluate the similarity of the input and trigger-embedded graphs. Besides, we apply the practical significance of features to constrain triggers.

The framework of our adaptive graph-level trigger generation method is shown in Figure 2. In the following, we elaborate on each key component.

C. Proportional Random Node Selection

Existing graph backdoor attacks on graph-level tasks usually adopt a static number of nodes to set triggers, which have several limitations. As far as we know, existing graph backdoor attacks use three to five nodes as triggers [18], [19], [20], which may not be able to achieve a high ASR on graphs with hundreds or even thousands of nodes. Besides, if the number of nodes in a graph is small, the trigger is easily detected if the topology of the trigger-embedded nodes is changed. In addition, to achieve a high ASR, the existing graph backdoor attacks would choose some important nodes (e.g., with a high degree) for triggering, leading to easy detection and defense.

We design an entirely random, dynamic, and topology-free node selection mechanism. Unlike existing attacks, in order to ensure the randomness of triggers, we completely randomly select trigger nodes without considering the importance of the nodes or the topology of the nodes. Additionally, the number of trigger nodes is determined based on the node size of the graph sample. This approach enhances the effectiveness and evasiveness of the attack since the number of trigger nodes appears natural relative to the graph size. Thus, the trigger of each malicious sample could be defined as follows:

$$g_i^t = \mathcal{R}_G\left(G_i, \alpha\_M_i\right), \tag{7}$$

where $\mathcal{R}_G$ is our proportional random node selection method, $\alpha\_M_i = \lceil \alpha M_i \rceil$, and $\lceil \cdot \rceil$ represents the rounding up function, α is our proportional parameter and M_i is the number of nodes of G_i as mentioned before.

D. Feature-based Trigger

There are two key factors in subgraph trigger generation: the topology of the subgraph trigger and the node features of the subgraph trigger nodes. Existing graph-level backdoor attacks against GNNs could be divided into two categories: one modifies only the topology of subgraph triggers, and the other modifies the topology and node features of subgraph triggers.

After evaluating the impact of modifying trigger nodes' topology, we found that just changing the topology may not achieve a high success rate, e.g., if the size of nodes in a graph is hundreds or thousands, a practical attack could not be achieved by using only three or four nodes as triggers. Besides, it is easy to detect in some cases, e.g., when a graph has only a few or a dozen nodes, using three or four nodes as triggers could be easily detected, and the slightest change in its connectivity would be noticeable.

The features of nodes usually have an essential impact on the prediction results of GNN models. Only modifying trigger node features could maintain the original graph's structural integrity, thus reducing the risk of detection. Furthermore, only modifying the trigger node features reduces the cost and complexity of the attack. As a result, our focus is on feature modification without altering the trigger nodes' topology.

Based on the preceding analysis, we propose a feature-based trigger generation method to achieve a high ASR and undetectability. The design of this dynamic trigger involves the following steps: (1) to make the distribution of the trigger nodes' feature values look less abnormal, we consider taking the mean of node features as the baseline; (2) we incorporate different offsets for different triggers across different samples to enhance the diversity of triggers.

For a graph sample G_i, we would modify the feature $x_i^{j,k} (1 \leq k \leq d, v_i^j \in \boldsymbol{V}_i^t)$ of the target node v_i^j to the same value. It is necessary to adjust the offset of the initial feature value by either increasing or decreasing it to create a noticeable difference from the original value. This modification allows the model to learn the trigger pattern effectively, achieving a successful attack. This paper focuses on increasing the feature value to accomplish this goal. To formalize this feature modification process, we define a feature modification formula:

$$\max \ \frac{1}{M_i} \sum_{q=0}^{M_i} x_i^{q,k} + \varphi, \tag{8}$$

where x_i^k denotes the vector of the k-th feature of all target nodes and φ represents the offset of trigger node features.

E. Constraints

We provide three trigger feature constraints to ensure the rationality of trigger feature values. The three constraints are as follows:

Similarity constraint. We compute the graph similarity between the trigger-embedded and original graphs using cosine similarity. We do not modify the subgraph triggers' topology but modify the nodes' features. Therefore, we consider using the cosine similarity of the trigger node features before and after the change to constrain the trigger. We define similarity thresholds T_G to ensure the rationality of malicious samples with embedded triggers:

$$\begin{aligned} \operatorname{sim}\left(G_i, G_{i,g_t}\right) &= \frac{\left\langle \boldsymbol{X}_i, \boldsymbol{X}_{i,g_t} \right\rangle}{\left|\left|\boldsymbol{X}_i\right|\right| \cdot \left|\left|\boldsymbol{X}_{i,g_t}\right|\right|} \\ &= \frac{\left[\boldsymbol{x}_i^1, \cdots, \boldsymbol{x}_i^{M_i}\right] \cdot \left[\boldsymbol{x}_{i,g_t}^1, \cdots, \boldsymbol{x}_{i,g_t}^{M_i}\right]}{\left|\left[\boldsymbol{x}_i^1, \cdots, \boldsymbol{x}_i^{M_i}\right]\right| \times \left|\left[\boldsymbol{x}_{i,g_t}^1, \cdots, \boldsymbol{x}_{i,g_t}^{M_i}\right]\right|} > T_G, \end{aligned} \tag{9}$$

where $[\cdot]$ is the concatenation operation, $\boldsymbol{X}_{i,g_t}$ represents the set of the node features of the trigger-embedded graph of G_i.

Range constraint. We perform statistical analysis on the nodes designated for modification, extracting their minimum and maximum feature values. Subsequently, we apply constraints to the feature values of the trigger nodes to prevent obvious outliers according to the modified node feature values:

$$\min x_i^{q,k} \leq \frac{1}{M_i} \sum_{q=0}^{M_i} x_i^{q,k} + \varphi \leq \max x_i^{q,k}. \tag{10}$$

Numeric constraint. We analyze the practical significance of the feature value of the node and correct it. For instance, if the feature value of the node is the atomic number, it could only be an integer. If the modified value is a fractional number, it is an obviously wrong value. Thus, we need to correct it. We use $\tau\left(x_i^{j,k}\right)$ to represent the type of $x_i^{j,k}$, then the constraint can be described as:

$$\tau\left(\frac{1}{M_i} \sum_{q=0}^{M_i} x_i^{q,k} + \varphi\right) = \tau\left(x_i^{j,k}\right). \tag{11}$$

We can offset the feature values of the trigger node by a small or large value to distinguish it from the original feature, ensuring a high success rate of the attack. In this paper, we choose to offset it to a larger value. Considering all the three constraints, our problem could be defined as:

$$\begin{aligned} \max \quad & \frac{1}{M_i} \sum_{q=0}^{M_i} x_i^{q,k} + \varphi \\ s.t. \quad & \operatorname{sim}(G_i, G_{i,g_t}) > T_G, \\ & \frac{1}{M_i} \sum_{q=0}^{M_i} x_i^{q,k} + \varphi \leq \max x_i^{q,k}, 1 \leq q \leq M_i, \\ & \tau\left(\frac{1}{M_i} \sum_{q=0}^{M_i} x_i^{q,k} + \varphi\right) = \tau\left(x_i^{j,k}\right), \\ & 1 \leq i \leq N_t, \\ & v_i^j \in \boldsymbol{V}_i^t. \end{aligned} \tag{12}$$

Algorithm 1 describes the process of our adaptive subgraph trigger generation method.

Algorithm 1: Adaptive Subgraph Trigger Generation.

Input: Dataset the adversary can access $\boldsymbol{D}_t$, number of samples N_t, proportional parameter α, threshold of graph similarity T_G, the number of node features d.

Output: Malicious dataset $\boldsymbol{D}_t$.

for $(G_i, y_i) \in \boldsymbol{D}_t$ **do**
  // selecting nodes randomly as subgraph trigger
  $g_i^t \leftarrow \mathcal{R}_G(G_i, \alpha\_M_i)$;
  // initializing features of each trigger node
  for $j = 1, 2, \cdots, \alpha\_M_i$ **do**
    for $k = 1, 2, \cdots, d$ **do**
      $x_{i,g_t}^{j,k} \leftarrow \frac{1}{M_i} \sum_{q=0}^{M_i} x_i^{q,k}$;
  Find optimized offset according to constraints;
  // modifying the feature
  for $j = 1, 2, \cdots, \alpha\_M_i$ **do**
    for $k = 1, 2, \cdots, d$ **do**
      $x_{i,g_t}^{j,k} \leftarrow x_{i,g_t}^{j,k} + \varphi$;
return $\boldsymbol{D}_t$;

F. An Illustrative Example

For example, in Figure 3, the i-th graph of a molecular structure dataset to be attacked by us contains ten nodes. Each node represents an atom and has four features $\boldsymbol{x}_i^j = \left[x_i^{j,1}, x_i^{j,2}, x_i^{j,3}, x_i^{j,4}\right]$, which represents the atomic number, valence and its position in the two-dimensional space, respectively. Firstly, we randomly select $\alpha = 20\%$ nodes as subgraph triggers, $g_i^t = \mathcal{R}_G(G_i, \alpha M_i)$, v_i^4, v_i^6 are selected as the trigger nodes. It is noted that the proportion could be less than 20% when the number of nodes is large. Secondly, we modify the features of these nodes. Assuming that we modify the first node feature of nodes v_i^4, v_i^6. We first calculate the average value: $\frac{1}{M_i}\sum_{q=0}^{M_i} x_i^{q,k} = \frac{1}{10}\sum_{j=1}^{10} x_i^{j,1} = 1.7$; then we modify the trigger nodes' first feature $x_{i,g_t}^{4,1} = x_{i,g_t}^{6,1} = \frac{1}{M_i}\sum_{q=0}^{M_i} x_i^{q,k} = 1.7$ as the initial value, we apply the same modification to all features.

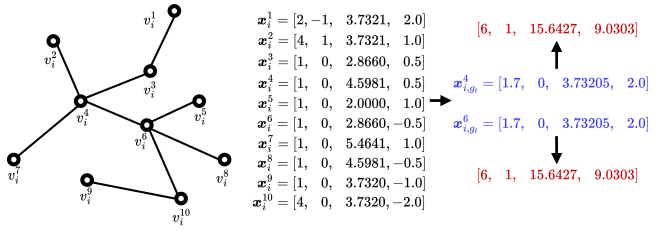


Fig. 3. An example of our subgraph trigger generation.

Then, we offset the initial value of each trigger node's features according to the defined constraints. $\max\{x_i^{q,1}\} = 10, \max\{x_i^{q,2}\} = 1, \max\{x_i^{q,3}\} = 25.3827, \max\{x_i^{q,4}\} = 15.5265$, and $T_G = 0.7$. According to the constraints, we use a greedy algorithm to find the optimal offsets, and we modify the trigger node features as $\boldsymbol{x}_{i,g_t}^4 = \boldsymbol{x}_{i,g_t}^6 =$ [6, 1, 15.6427, 9.0303].

After modifying the features, the similarity between G_i and G_{i,g_t} is computed as $\text{sim}(G_i, G_{i,g_t}) = 0.7033$.

V. Adaptive Node-level Backdoor Attack

Node-level tasks in GNNs are pivotal across various domains, like article classification in citation networks [37] and user preference judgment in recommendation systems [38]. However, the threat of backdoor attacks poses a significant concern. Backdoor attacks can manipulate GNNs to misclassify documents and generate inaccurate recommendations, potentially leading to misinformation, hindering knowledge discovery, and undermining the entire system's credibility.

A. Attack Analysis

For GNN node-level tasks, the effectiveness of backdoor attacks depends on recognizing node triggers and the difference in vector representations between nodes with and without embedded triggers. When node triggers have high recognition, and the vector representations of nodes with and without embedded triggers are significantly different, training the model using nodes with embedded triggers can enable the model to learn the characteristics of triggers better, i.e., classify nodes with embedded triggers into one category, which is the target category of the adversary. Assume that a trigger is added to v^i, and its features are modified to $\boldsymbol{x}_{g_t}^i$, δ is the successful attack threshold. The difference between the vector representation of the node embedded with the trigger and its corresponding node without the trigger embedded in the output layer is:

$$\Delta_{\boldsymbol{h}_{v^i}} = \left|\left|\boldsymbol{h}_{v_{g_t}^i} - \boldsymbol{h}_{v^i}\right|\right|, \tag{13}$$

when $\Delta_{\boldsymbol{h}_{v^i}} > \delta$, the backdoor can be successfully embedded in the model, and the model embedded with the backdoor can misclassify the node embedded with the trigger as the target category of the adversary. Since GNN needs to aggregate the vector representation of the node and its neighbor nodes in the previous layer when updating the vector representation of each node in the input layer and the hidden layer, this operation will weaken the characteristics of the node features. Therefore, in order to ensure that $\Delta_{\boldsymbol{h}_{v^i}} > \delta$, three aspects can be considered: 1) offset the original features of the target node as much as possible; 2) construct a new special node and connect it to the target node to strengthen the model's recognition of the trigger; 3) you can also consider pruning the edges between the target node and its neighbor nodes to enhance the model's learning effect on the trigger.

The evasiveness of the backdoor attack depends on whether the attack significantly changes the overall feature distribution of the node features. If the feature distributions of nodes without embedded triggers and nodes with embedded triggers are similar, the attack is considered to be hidden, and the cosine similarity can be used to evaluate the similarity between them:

$$\text{sim}(v^i, v_{g_t}^i) = \frac{\left\langle \boldsymbol{x}^i, \boldsymbol{x}_{g_t}^i \right\rangle}{\left|\left|\boldsymbol{x}^i\right|\right| \cdot \left|\left|\boldsymbol{x}_{g_t}^i\right|\right|}, \tag{14}$$

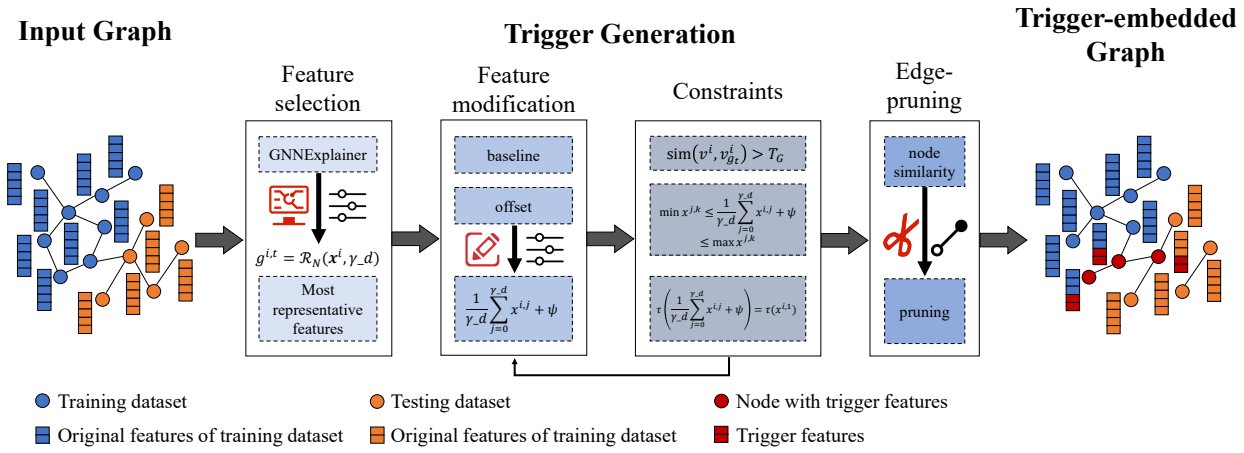


Fig. 4. The framework of our adaptive node-level trigger generation method.

$\boldsymbol{x}^i$ and $\boldsymbol{x}^i_{g_t}$ are the node features without embedded triggers and nodes with embedded triggers, respectively.

B. Attack Overview

We use $G = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{X})$ to represent a graph, where $\boldsymbol{V} = \{v^1, v^2, \cdots, v^M\}$ is the set of M nodes, $\boldsymbol{E} \subseteq \boldsymbol{V} \times \boldsymbol{V}$ is the set of edges, and $\boldsymbol{X} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^M\}$ is the set of node features with $\boldsymbol{x}^i = \{x^{i,1}, x^{i,2}, \cdots, x^{i,d}\}$ being the node feature of v_i, where d is the dimension of the node features. $\boldsymbol{A} \in \mathbb{R}^{M \times M}$ is the adjacency matrix of the graph G, where $\boldsymbol{A}^{i,j} = 1$ if nodes v^i and v^j are connected; otherwise $\boldsymbol{A}^{i,j} = 0$. In this section, we focus on a node classification task in the inductive setting, which widely exists in real-world applications. For instance, GNNs trained on social networks often need to conduct predictions on newly enrolled users to provide service. Specifically, in inductive node classification, the labels of all nodes $\boldsymbol{V}$ is $\boldsymbol{y} = \{y_1, y_2, \cdots, y_M\}$. The test nodes $\boldsymbol{V}_T$ are not covered in the training graph G, i.e., $\boldsymbol{V}_T \cap \boldsymbol{V} = \emptyset$.

For a node sample v^i, we use the subset of the node features $\boldsymbol{g}^{i,t} \subseteq \boldsymbol{x}^i$ as its trigger. Assuming that we could access a subset $\boldsymbol{V}_t \subseteq \boldsymbol{V}$, the number of the node samples of $\boldsymbol{V}_t$ is N_t. In order to get a high success rate and undetectability of the attack, the training process of the target model θ can be defined as:

$$\theta_t = \underset{\theta}{\operatorname{argmin}} \left(\sum_{v^i \in \boldsymbol{V}_c} \ell\left(\theta(v^i), y_i\right) + \sum_{v^i \in \boldsymbol{V}_t} \ell\left(\theta\left(\mathcal{A}_N\left(v^i\right)\right), y_t\right) \right), \tag{15}$$

where $\boldsymbol{V}_c = \boldsymbol{V} - \boldsymbol{V}_t$ and $\mathcal{A}_N(\cdot)$ is our node trigger generation method.

Similar to our graph-level backdoor attack, we explored a broader trigger generation approach for node-level tasks. We have to tackle three key **challenges** similar to the case of the graph-level backdoor attack: (1) *intricate trigger design*, (2) *reasonable constraint construction*, and (3) *efficient reinforcement strategy*.

To address the above challenges, we propose three new mechanisms as follows. (1) We use GNNExplainer to analyze the importance of each node feature and determine the trigger features according to the importance of node features and the size of the feature dimension. (2) We present a highly concealed feature modification method. We use cosine similarity to evaluate the similarity of the input node and the trigger-embedded node and apply the practical significance of features to constrain triggers. (3) We have devised an adaptive edge-pruning mechanism to achieve a notably high ASR.

The framework of our adaptive node-level trigger generation method is shown in Figure 4. In the following, we elaborate on each key component.

C. Feature Selection

When we perform backdoor attacks on node-level tasks, we modify node features as triggers. In order to ensure a high ASR and undetectability of our method, we consider the adaptive selection of the features of the nodes to be attacked. We use the popular GNNExplainer to analyze the importance of the features of each node.

According to the working mode of GNNs, the embedding of a node would be generated by a subgraph G_c of the whole graph G. GNNExplainer's goal is to identify a subgraph $G_s \subseteq G_c$ and the associated masked features F_s^m that are important for the GNN's prediction. The framework of GNNExplainer could be described as:

$$\max_{G_s, m} \mathrm{MI}\left(y_t, (G_s, m)\right) = H\left(y_t\right) - H\left(y_t \mid G = G_s, F = F_s^m\right), \tag{16}$$

where MI is mutual information.

After analyzing the importance of features of each node, for i-th malicious sample, we select the *top*-γ_d important features as a trigger, where $\gamma\_d = \lceil \gamma d \rceil$, and $\lceil \cdot \rceil$ represents the rounding up function:

$$g^{i,t} = \mathcal{R}_N\left(\boldsymbol{x}^i, \gamma\_d\right), \tag{17}$$

where $\mathcal{R}_N$ is our feature selection method, γ denotes a proportional parameter.

D. Feature Modification

Our objective is that when the backdoored model recognizes that there are several features with the same value in the

node sample, the backdoor is triggered, and the sample is misclassified as the target class of the adversary. As a result, the key is how to modify the most important features we selected for the target nodes. In order to ensure the rationality of trigger feature values, we consider taking the mean of the selected features as the basis. In addition, in order to ensure a high ASR and diversity of triggers, we consider different offsets for triggers of different samples. For a node sample v^i, we use $v^i_{g_t}$ to represent the trigger-embedded node sample and $\boldsymbol{x}^i_{g_t}$ represent the trigger-embedded features. We would modify the top-γ_d important features $\boldsymbol{x}^i_{g_t} = \left[x^{i,1}_{g_t}, x^{i,2}_{g_t}, \cdots, x^{i,\gamma\_d}_{g_t}\right]$ to the same value $x^i_{g_t}$, we define a feature modification formula:

$$\max \ \frac{1}{\gamma\_d}\sum_{j=0}^{\gamma\_d} x^{i,j} + \psi, \tag{18}$$

where ψ represents the offset of feature.

Similar to our graph-level attack, we consider increasing the feature value so as to have a more guaranteed effect on the model.

E. Constraints

Similarly to the graph-level attack, we provide three trigger feature constraints to ensure the rationality of trigger feature values. The three constraints are as follows:

Similarity constraint. We compute the node similarity between the trigger-embedded node and the original node using cosine similarity (Cosim). We define similarity thresholds T_N to ensure the rationality of malicious samples with embedded triggers:

$$\begin{aligned}\text{sim}\left(v^i, v^i_{g_t}\right) &= \frac{\left\langle \boldsymbol{x}^i, \boldsymbol{x}^i_{g_t}\right\rangle}{\left|\left|\boldsymbol{x}^i\right|\right| \cdot \left|\left|\boldsymbol{x}^i_{g_t}\right|\right|} \\ &= \frac{\sum_{j=1}^{d}\left(x^{i,j} \times x^{i,j}_{g_t}\right)}{\sqrt{\sum_{j=1}^{d}\left(x^{i,j}\right)^2} \times \sqrt{\sum_{j=1}^{d}\left(x^{i,j}_{g_t}\right)^2}} > T_N.\end{aligned} \tag{19}$$

Range constraint. We statistically analyze the features of the nodes to be modified and obtain the minimum and maximum values so as to constrain the feature values of the trigger nodes and avoid outliers in the modified node feature values.

$$\min x^{j,k} \le \frac{1}{\gamma\_d}\sum_{j=0}^{\gamma\_d} x^{i,j} + \psi \le \max x^{j,k}, 1 \le k \le d, v^j \in \boldsymbol{V}_t. \tag{20}$$

Numeric constraint. We analyze the practical significance of the feature values of the node and correct it. We use $\tau\left(x^i_{g_t}\right)$ to represent the type of $x^i_{g_t}$, then the constraint can be described as:

$$\tau\left(\frac{1}{\gamma\_d}\sum_{j=0}^{\gamma\_d} x^{i,j} + \psi\right) = \tau\left(x^{i,1}\right). \tag{21}$$

As for our graph-level backdoor attack, we choose to offset trigger features to a larger value. Considering all the three constraints, our problem could be defined as:

$$\begin{aligned}\max \quad & \frac{1}{\gamma\_d}\sum_{j=0}^{\gamma\_d} x^{i,j} + \psi \\ \text{s.t.} \quad & \text{sim}\left(v^i, v^i_{g_t}\right) > T_N, \\ & \frac{1}{\gamma\_d}\sum_{j=0}^{\gamma\_d} x^{i,j} + \psi \le \max x^{j,k}, 1 \le k \le d, \\ & \tau\left(\frac{1}{\gamma\_d}\sum_{j=0}^{\gamma\_d} x^{i,j} + \psi\right) = \tau\left(x^{i,1}\right), \\ & v^j \in \boldsymbol{V}_t.\end{aligned} \tag{22}$$

F. Adaptive Edge-Pruning

When updating a node's embedding, GNNs achieve this by aggregating the embeddings of the node and its neighbors. However, simply modifying node characteristics as a trigger is insufficient for effective attacks due to the effect of neighbor nodes. Some existing methods employ subgraph modifications, connecting them to the target node as triggers or introducing new neighbor nodes. These approaches aim to amplify the trigger feature's impact, yet they can be easily countered by the defense method RS.

To address this, we propose a novel edge-pruning approach. We traverse the neighbor nodes of trigger-embedded node samples. When the similarity between one of its neighbor nodes and the adversary's target node is greater than a new similarity threshold $T_S = 0.5$, we prune the edges between the malicious node sample and the neighbor node. Assuming that the adversary wants to attack node v^i:

$$\boldsymbol{A} = \left[\boldsymbol{A}^{i,j} | \boldsymbol{A}^{i,j} = \begin{cases} 0, & \text{if sim}(v^i, v^j) < T_S \\ \boldsymbol{A}^{i,j}, & \text{others} \end{cases}\right], \ v^j \in \mathcal{N}_i, \tag{23}$$

where $\boldsymbol{A}^{i,j}$ represents the connection relationship of nodes v^i and v^j, and $\mathcal{N}_i$ are the set of neighbors of node v^i.

Algorithm 2 describes the process of our adaptive node trigger generation method. Overall, our node-level trigger generation method takes into account the randomness, diversity, scalability, and rationality of the feature values of the malicious samples.

G. An Illustrative Example

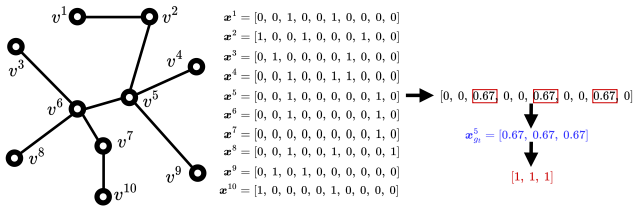


Fig. 5. An example of our node trigger generation.

For example, Figure 5 is a citation network's subgraph that we can access. Each node represents an article, and

Algorithm 2: Adaptive Node Trigger Generation.

Input: Subset the adversary can access $\boldsymbol{V}_t$, number of samples N_t, proportional parameter γ, dimension of node features d, threshold of node similarity T_N, threshold of edge-pruning T_S.

Output: Malicious subset $\boldsymbol{V}_t$.

for $v^i \in \boldsymbol{V}_t$ **do**
  // selecting most important features as trigger
  $g^{i,t} \leftarrow \mathcal{R}_N\left(\boldsymbol{x}^i, \gamma\_d\right)$;
  // initialize node trigger
  $x^i_{g_t} \leftarrow \frac{1}{\gamma\_d} \sum_{j=0}^{\gamma\_d} x^{i,j}_{g_t}$;
  Find optimized offset according to constraints;
  // modifying the features
  for $j = 1, 2, \cdots, \gamma\_d$ **do**
    $x^{i,j}_{g_t} \leftarrow x^i_{g_t} + \psi$;
  // Adaptive edge-pruning.
  for $\forall v^j \in \mathcal{N}_i$ **do**
    if $sim\left(v^i, v^j\right) < T_S$ **then**
      $\boldsymbol{A}^{i,j} \leftarrow 0$;
return $\boldsymbol{V}_t$;

each feature represents whether the article contains a certain keyword. Zero means the article does not contain a certain word, and one means it does. Each node has ten features $\boldsymbol{x}^i = \{x^{i,1}, x^{i,2}, \cdots, x^{i,10}\}$. We would attack the node v^5. Firstly, we analyze the feature importance of v^5 according to GNNExplainer and set $\gamma = 0.3$. We select the top-γ_d important features as a trigger, $g^{i,t} = \mathcal{R}_N\left(\boldsymbol{x}^i, \gamma\_d\right)$. Secondly, we modify the trigger features. We first calculate the average value: $\frac{1}{\gamma\_d}\sum_{j=0}^{\gamma\_d} x^{i,j} = \frac{1}{3}\sum_{j=1}^{3} x^5_{g_t} = 0.67$; then we modify the features of node v^5 as: $\boldsymbol{x}^5 = [0, 0, 0.67, 0, 0, 0.67, 0, 0, 0.67, 0]$ as the initial values.

After we initialize the values of trigger node features, we offset the trigger node features according to the defined constraints. $\max x^{j,k} = 1(1 \le k \le d, v^j \in \boldsymbol{V}_t)$, and we set $T_N = 0.5$. According to the constraints, we use a greedy algorithm to find the optimal offsets, and we modify the features of node v^5 as $\boldsymbol{x}^5 = [0, 0, 1, 0, 0, 1, 0, 0, 1, 0]$. After we modify the features, we can compute the $\text{sim}\left(v^i, v^i_{g_t}\right) = 0.8165$.

After modifying the node features, we prune edges adaptively. We set $T_S = 0.5$ and compute the similarity of v^5 and its neighbors v^2, v^4, v^6, v^9: $\text{sim}\left(v^5, v^2\right) = 0$, $\text{sim}\left(v^5, v^4\right) = 0.67$, $\text{sim}\left(v^5, v^6\right) = 0.82$, $\text{sim}\left(v^5, v^9\right) = 0$. Therefore, we prune the edges of v^5 and v^2, v^5 and v^9.

VI. Performance Evaluation

A. Experiment Settings

Dataset. To demonstrate the effectiveness of our attack, we used four popular graph-structured data, AIDS [39], PROTEINS_full [40], Fingerprint [41], Cora [42], CiteSeer [43] and Flickr [44] to conduct our experiments. AIDS is a molecular structure graph of active and inactive compounds. PROTEINS_full is a dataset of proteins that are classified as enzymes or non-enzymes. The fingerprint is a dataset representing each fingerprint as a graph. The above three datasets are used for graph classification tasks. Cora and CiteSeer are small citation networks, and they are used for node classification tasks. Flickr is a large-scale graph that connects image captions with the same attributes, and it is also used for node classification tasks. The statistics of the datasets are summarized in Table II.

Reasons for choosing datasets: The chosen datasets are widely used benchmarks in graph learning, covering a range of applications from bioinformatics to pattern recognition. These datasets ensure that the evaluation results are relevant and comparable to other studies in the literature.

AIDS provides a diverse range of molecular graphs, which helps test the robustness and generalization of the proposed method across different biological molecules. In PROTEINS_full, the complexity and variety of protein structures make it an excellent benchmark for assessing the method's ability to handle complex and heterogeneous graph data. Fingerprint offers a real-world application scenario, testing the method's performance in practical and high-stakes environments.

Cora and CiteSeer are standard benchmarks for testing node classification tasks in graph learning, providing a clear and well-defined task to evaluate the method's performance. Flickr is a large-scale graph testing the effectiveness of ABARC against large-scale graphs.

Dataset splits and parameter setting. For all datasets we used, we randomly selected 80% of the samples as the training set and the remaining 20% as the test set.

Models. In our evaluation, we used three SOTA GNN models: GCN [4], GraphSAGE [3], [29] and GAT [30]. Using GNNs of distinct network architectures (i.e., graph convolution, general aggregation function, versus graph attention), we factor out the influence of the characteristics of individual models.

Reasons for choosing models: The chosen GNN models represent a broad spectrum of graph neural network architectures, from basic convolutional approaches to attention-based and scalable inductive methods. This diversity in models ensures comprehensive evaluation across different types of GNN architectures.

GCN, with its widespread adoption and proven performance in various graph-based tasks, serves as a robust baseline. The use of GAT as an evaluation model allows us to demonstrate how the proposed method performs with models that incorporate advanced attention mechanisms for more nuanced graph representation learning. The scalability and inductive capabilities of GraphSAGE are crucial for evaluating the proposed method's performance on large-scale graphs and in dynamic environments where the graph structure may evolve.

Baselines. For graph classification tasks, we use four baselines: the non-backdoored model Benign, GTA [20], GTA-t, the variants of GTA, which only optimizes the trigger's topological connectivity, and BKD [18]. For node classification tasks, we use the non-backdoored model Benign, EXP [19], GTA, and UGBA [21] as baselines.

TABLE II
DATASET STATISTICS.

Dataset	#(graphs)	$\overline{\#(\text{nodes})}$	$\overline{\#(\text{edges})}$	#(classes)	#(graphs) (class label)	target class
AIDS	2,000	15.69	16.20	2	400[0], 1,600[1]	0
PROTEINS_full	1,113	39.06	92.14	2	663[0], 450[1]	0
Fingerprint	1,661	8.15	6.81	4	538[0], 517[1], 109[2], 497[3]	0
Cora	1	2,708	5,429	7	351[0], 217[1], 418[2], 818[3], 426[4], 298[5], 180[6]	0
CiteSeer	1	3,327	4,608	6	264[0], 590[1], 668[2], 701[3], 596[4], 508[5]	0
Flickr	1	89,250	899,756	7	2,628[0], 4,321[1], 3,164[2], 2,431[3], 11,525[4], 1,742[5], 18,814[6]	0

Reasons for choosing baselines: The selected baseline attack methods are the most advanced GNN backdoor attacks. By evaluating and comparing the attack performances of ABARC and these advanced attacks, we can effectively verify the effectiveness and evasiveness of ABARC, underscoring the significance of our research.

Metrics. To evaluate attack effectiveness, we use ASR, which measures the likelihood that the backdoored model θ_t classifies trigger-embedded trials to the target class designated by the adversary:

$$\text{ASR} = \frac{\#(\text{successful trigger-embedded trials})}{\#(\text{total trials})}\bigg|_{\theta_t} \tag{24}$$

To evaluate the attack evasiveness, we use CAD, which measures the difference in classification accuracy of non-backdoored model θ and its trojan counterpart θ_t with respect to non-trigger-embedded trials:

$$\begin{aligned}\text{CAD} = &\frac{\#(\text{successful non-trigger-embedded trials})}{\#(\text{total trials})}\bigg|_{\theta} \\ - &\frac{\#(\text{successful non-trigger-embedded trials})}{\#(\text{total trials})}\bigg|_{\theta_t}\end{aligned} \tag{25}$$

Defense. We consider mitigating and detecting our attack from aspects of the model and the input. We investigate the effectiveness of two SOTA defense methods in mitigating and detecting our backdoor attack:

Input-inspection: Randomized Smoothing (RS) [18]. RS applies a subsampling function over a given graph, generates a set of subsamples, and takes a majority voting of the predictions over such subsamples as the graph's final prediction.

Model-inspection: Neural Cleanse (NC) [45]. NC assumes that a specific class is the target class of the adversary, then adds the same perturbation to all non-target class samples and looks for the minimum perturbation that allows the model to classify all non-target class samples into the target class. NC generates a minimum perturbation for each class and determines whether the model contains a backdoor by comparing whether there are outliers in the generated perturbations.

Reasons for choosing defense methods: RS operates at the input level, applying subsampling and majority voting to enhance the robustness of the model's predictions against adversarial perturbations. By manipulating the input data, RS aims to reduce the impact of any backdoor-triggering patterns that may be present in the input graph, thereby mitigating the effect of the backdoor attack. NC, on the other hand, focuses on the model itself. This model-centric approach helps detect the presence of backdoor triggers embedded within the model's parameters, providing a mechanism to identify and analyze suspicious behavior.

RS's strength lies in its ability to enhance the model's resilience to input-based attacks, making it harder for adversarial examples to manipulate predictions. NC complements this by providing an in-depth analysis of the model's parameters and their susceptibility to backdoor triggers. Its ability to detect outliers in perturbation magnitudes helps identify and understand backdoor mechanisms that may not be apparent through input inspection alone.

Combining these two SOTA methods, we leverage their strengths to provide a comprehensive defense strategy that addresses backdoor attacks.

B. Results for Graph Classification

We assume that the adversary has access to only 1% of the complete dataset. During the test phase, we strategically choose to add triggers to 25% of the samples in the test set.

Attack efficacy. When evaluating our experimental results, according to the attack performance of various α and T_G mentioned later, we set $\alpha = 0.2$ and $T_G = 0.5$. The outcomes of our experiments are presented in Table III. The table clearly illustrates the performance of ARBC and other attacks, as our ASR reaches over 96.43%, while the CAD remains below 0.64%. Additionally, it is worth noting that the GTA-t attack fails to achieve significant attack effectiveness, highlighting the inadequacy of the subgraph trigger topology.

We applied the input-inspection backdoor mitigating strategy RS to defend against backdoor attacks. We define a parameter β, which means we randomly remove $\beta \times 100\%$ nodes of each graph sample, and for the rest nodes, we randomly set $\beta \times 100\%$ of their features to be 0. The larger β is, the better RS can mitigate the impact of backdoor attacks. Figure 6 shows the performance of Benign as β varies from 0.0 to 0.8. We can observe that when $\beta = 0.4$, the Benign model can maintain a good performance while mitigating backdoor attacks to the greatest extent. Therefore, we set $\beta = 0.4$. Experimental results illustrate the robustness of our attack, which maintains a high ASR of approximately 96% with only a marginal 2% drop, while other attacks experience a significant 10% to 30% drop in their ASR.

We also apply the model-inspection backdoor defense strategy NC to evaluate the robustness of our graph-level backdoor

TABLE III
BACKDOOR ATTACK RESULTS OF GRAPH-LEVEL TASKS (ASR (%) | CAD (%)).

Dataset	Defense	Model	Benign	GTA	GTA-t	BKD	ABARC (Ours)
AIDS	None	GCN	96.50	99.00 | 0.17	48.00 | 10.50	98.00 | 0.42	**99.00** | **0.00**
		GAT	95.75	99.00 | 1.42	50.00 | 9.75	**100.00** | **0.00**	97.00 | 0.08
		GraphSAGE	95.75	99.00 | 0.42	62.00 | 25.75	99.00 | 0.16	**99.00** | **0.08**
	RS	GCN	95.75	87.00 | 7.08	48.00 | 8.08	75.00 | 5.42	**94.00** | **1.84**
		GAT	88.50	86.00 | 5.83	49.00 | 4.83	78.00 | 2.17	**95.00** | **1.83**
		GraphSAGE	91.75	79.00 | 3.42	41.00 | 0.42	78.00 | 0.00	**94.00** | **1.08**
PROTEINS_full	None	GCN	75.34	**100.00** | **0.49**	82.14 | 2.88	98.21 | 0.34	98.21 | 0.00
		GAT	72.65	98.21 | 1.39	78.57 | 3.34	98.21 | 0.00	**100.00** | **0.12**
		GraphSAGE	73.54	98.21 | 1.08	76.79 | 2.43	**98.21** | **0.43**	96.43 | 0.64
	RS	GCN	71.30	89.29 | 2.44	66.07 | 3.04	69.64 | 3.04	**96.43** | **1.84**
		GAT	66.37	71.43 | 0.92	69.29 | 5.11	60.71 | 0.00	**98.21** | **0.00**
		GraphSAGE	68.61	80.36 | 3.34	44.64 | 1.73	76.79 | 2.32	**96.43** | **0.00**
Fingerprint	None	GCN	82.83	97.59 | 0.09	84.33 | 1.62	93.98 | 1.91	**98.80** | **0.00**
		GAT	80.42	98.80 | 0.42	81.93 | 2.26	96.39 | 0.12	**100.00** | **0.10**
		GraphSAGE	81.33	**100.00** | **0.08**	84.34 | 1.32	96.39 | 0.23	98.80 | 0.21
	RS	GCN	79.82	87.95 | 1.22	72.29 | 2.23	69.88 | 1.38	**96.39** | **0.84**
		GAT	78.92	84.34 | 0.83	71.08 | 3.42	66.27 | 0.67	**95.18** | **0.05**
		GraphSAGE	79.22	83.13 | 2.24	80.72 | 2.27	69.88 | 1.57	**93.98** | **0.00**

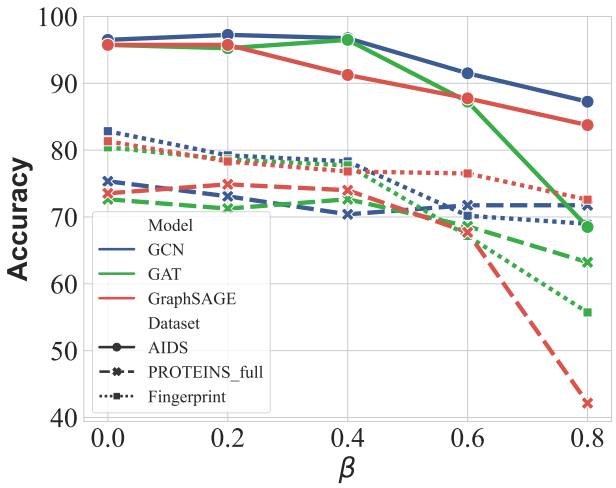


Fig. 6. The performance of Benign under various β of RS on graph-level tasks.

attack. When performing NC detection, we assume that a specific class is the target class of the adversary and then add the same perturbation to the features of each node of all non-target class graph samples, looking for a model to classify all non-target class graph samples as a minimal perturbation of the target class. We generate a minimum perturbation for each class and determine whether the model contains a backdoor by comparing whether there are outliers in the generated perturbations. The results are shown in Figure 7. We can observe that BKD shows significant differences across the two classes. At the same time, the results of Benign and ours seem the most similar, implying the robustness of our graph-level backdoor attack.

Trigger node proportion α. We conducted experiments by setting $T_G = 0.5$. Figure 8 depicts the performance of our method as α varies from 0.1 to 0.5. Notably, our attack displays robustness against changes in α, with minimal impact on ASR and CAD. This resilience could be attributed to the fact that the dataset comprises graphs of varying sizes, and we ensure the adaptiveness of our designed triggers by selecting them in proportion to the graph size.

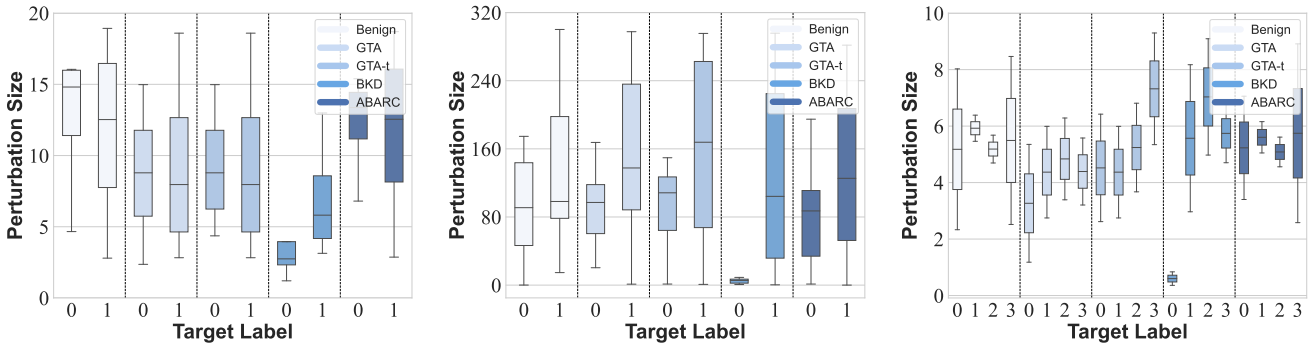
Similarity threshold T_G. Furthermore, we investigate the impact of the similarity threshold T_G by setting $\alpha = 0.2$. Figure 9 shows the performance of our method as the similarity threshold T_G ranges from 0.3 to 0.99. The experimental results indicate that as the similarity between the original graph and the trigger-embedded graph increases, the ASR exhibits a noticeable decreasing trend, while CAD demonstrates a distinct increasing trend. Essentially, a higher similarity implies that the trigger becomes more evident as we shift the features of the trigger node, resulting in a more pronounced attack effectiveness.

C. Results for Node Classification

We assume that the adversary has access to 5% of the entire dataset. At test time, we add triggers to 50% of the samples in the test set.

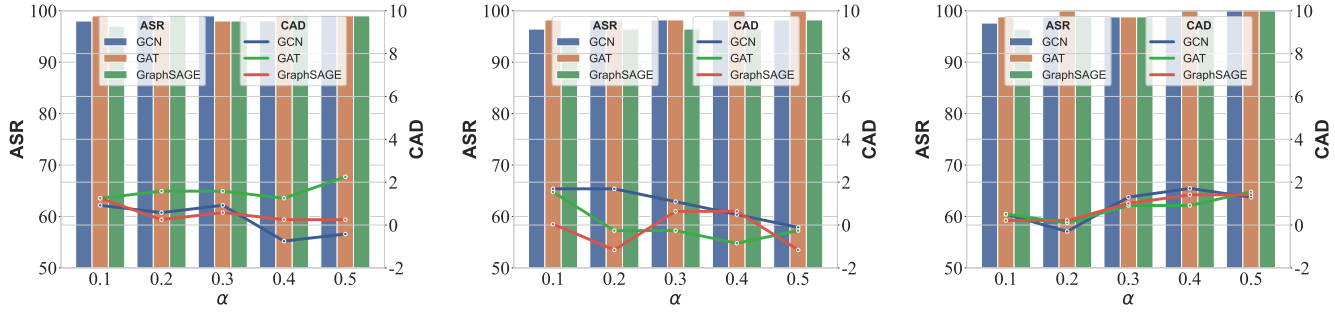
Attack efficacy. When evaluating our experimental results, according to the attack performance of various γ, T_N and T_S mentioned later, we set $\gamma = 0.3$, $T_N = 0.5$ and $T_S = 0.5$. The outcomes of our experiments are presented in Table IV. The table clearly illustrates the effectiveness of our method, as our ASR reaches over 95.94%, while the CAD remains below 1.85%.

We applied the input-inspection backdoor mitigating strategy RS to defend against backdoor attacks. Like our graph-



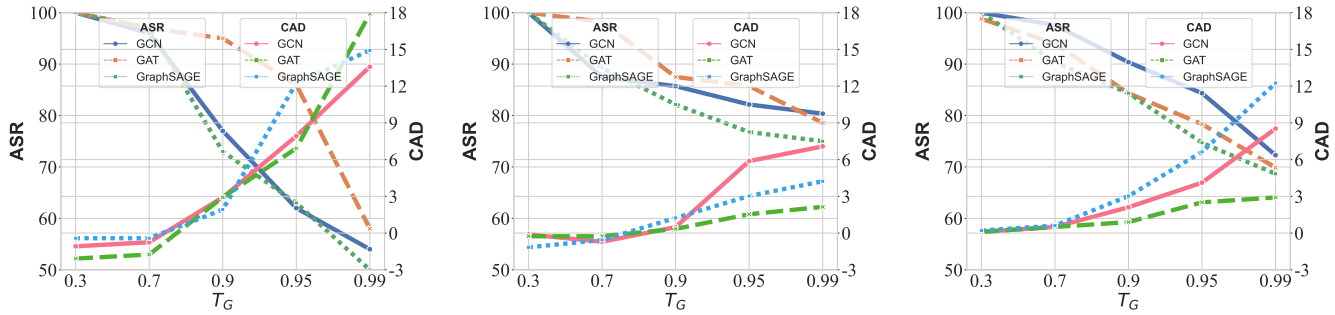
(a) AIDS (b) PROTEINS_full (c) Fingerprint

Fig. 7. Detection results of ABARC and other attacks by NC on the graph-level task using GCN.



(a) AIDS (b) PROTEINS_full (c) Fingerprint

Fig. 8. The attack performance of ABARC under various trigger node proportion α on graph-level tasks when $T_G = 0.5$.



(a) AIDS (b) PROTEINS_full (c) Fingerprint

Fig. 9. The attack performance of ABARC under various similarity T_G on graph-level tasks when $\alpha = 0.2$.

level backdoor attack, we define a parameter β, which means we randomly prune the edge of the neighbor nodes of each node and set $\beta \times 100\%$ of each node's features to be 0. We test the performance of Benign as β varies from 0.0 to 0.8, as shown in Figure 10. We can observe that when $\beta = 0.4$, the Benign model can maintain a good performance while mitigating backdoor attacks to the greatest extent. Therefore, we set $\beta = 0.4$. Our results illustrate the robustness of our method. Our attack maintains a high ASR of over 99% with no drop, while other attacks experience a significant 10% to 30% drop in their ASR.

We also applied the model-inspection backdoor detection defense strategy NC to evaluate the robustness of our node-level backdoor attack. When performing NC detection, we assume that a specific class is the target class of the adversary and then add the same perturbation to the features of each non-target class node, looking for a model to classify all non-target class node samples as a minimal perturbation of the target class. We generate a minimum perturbation for each class and determine whether the model contains a backdoor by comparing whether there are outliers in the generated perturbations. The results are shown in Figure 11. We can observe that for all node-level attacks, NC cannot detect the backdoor. The detection results of our model are very similar to Benign, which reflects the robustness of our node-level backdoor attack.

Feature trigger proportion γ. We set $T_N = 0.5$ and $T_S = 0.5$ to evaluate the impact of trigger node proportion

TABLE IV
BACKDOOR ATTACK RESULTS OF NODE-LEVEL TASKS (ASR (%) | CAD (%)).

Dataset	Defense	Model	Benign	EXP	GTA	UGBA	ABARC (Ours)
Cora	None	GCN	85.79	73.80 | 20.11	95.94 | 1.66	93.73 | 1.29	**95.94** | **0.74**
		GAT	86.16	74.54 | 20.48	96.31 | 3.13	94.83 | 2.77	**98.89** | **1.11**
		GraphSAGE	85.61	83.76 | 6.27	**99.26** | **0.74**	98.52 | 0.37	98.16 | 1.85
	RS	GCN	85.42	62.36 | 1.66	83.39 | 2.03	83.03 | 1.29	**100.00** | **0.55**
		GAT	84.31	63.10 | 2.39	80.44 | 0.00	73.80 | 0.00	**99.26** | **0.00**
		GraphSAGE	86.34	88.19 | 1.47	86.72 | 2.21	85.24 | 1.10	**99.63** | **0.00**
CiteSeer	None	GCN	76.73	40.54 | 7.36	96.40 | 3.16	96.70 | 4.66	**99.40** | **1.65**
		GAT	77.33	40.24 | 6.16	94.29 | 3.46	98.50 | 3.16	**100.00** | **0.75**
		GraphSAGE	76.58	51.95 | 1.50	**99.70** | **0.30**	94.89 | 0.60	99.70 | 1.81
	RS	GCN	75.98	35.14 | 1.21	86.79 | 2.71	80.48 | 2.71	**100.00** | **0.00**
		GAT	75.68	31.53 | 0.00	80.78 | 3.01	72.67 | 1.81	**100.00** | **0.00**
		GraphSAGE	76.58	55.56 | 0.60	89.19 | 0.00	81.98 | 2.11	**100.00** | **0.30**
Flickr	None	GCN	49.86	50.42 | 3.46	95.24 | 2.23	**97.06** | **0.82**	95.85 | 1.09
		GAT	49.77	44.82 | 3.18	96.72 | 2.64	97.84 | 4.23	**99.72** | **0.34**
		GraphSAGE	49.83	49.81 | 0.75	**99.98** | **0.23**	99.94 | 0.12	99.88 | 0.56
	RS	GCN	48.18	46.06 | 1.35	86.53 | 3.25	85.28 | 2.33	**100.00** | **0.00**
		GAT	47.62	39.45 | 0.67	84.27 | 1.13	73.06 | 0.87	**99.87** | **0.02**
		GraphSAGE	47.90	39.72 | 1.22	87.39 | 0.55	81.79 | 1.85	**100.00** | **0.00**

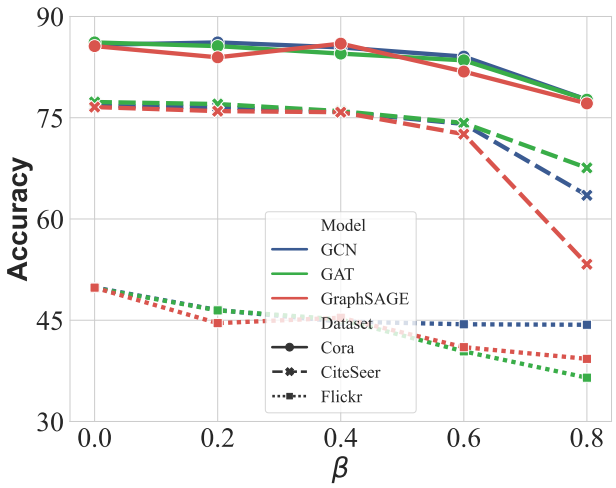


Fig. 10. The performance of Benign under various β of RS on node-level tasks.

γ on our attack. Figure 12 illustrates the performance of our method as γ varies from 0.01 to 0.7. The experimental results indicate that as the proportion of feature triggers increases, the ASR exhibits a noticeable increasing trend, while CAD demonstrates a distinct decreasing trend.

Similarity threshold T_N. We set $\gamma = 0.3$ and $T_S = 0.5$ to evaluate the impact of similarity threshold T_N on our attack. Figure 13 illustrates the performance of our method as the similarity threshold T_N varies from 0.1 to 0.9. Notably, our attack method displays robustness against changes in T_N, with minimal impact on ASR and CAD. This phenomenon could be attributed to the dataset's high number of features per node. By selectively choosing a subset of features as triggers, any modifications made to them would have a minimal impact on the similarity between nodes before and after the changes.

Adaptive edge-pruning threshold T_S. We set $\gamma = 0.3$ and $T_N = 0.5$ to evaluate the impact of edge-pruning threshold T_S on our attack. Figure 14 illustrates the performance of our node-level backdoor attack as T_S varies from 0.1 to 0.9. The experimental results indicate that as the threshold increases, the ASR exhibits a noticeable increasing trend, while CAD demonstrates a distinct decreasing trend.

D. The Performance of ABARC with Varying Graph Sizes

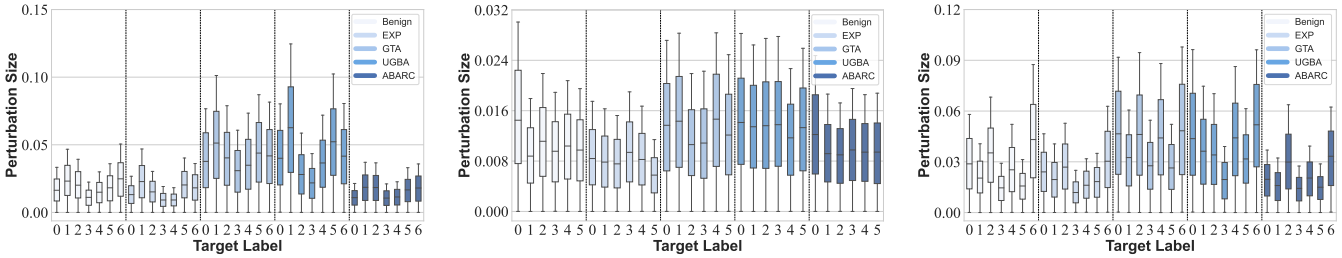
The performance of ABARC has been thoroughly evaluated across a diverse set of graph samples varying significantly in size and complexity. The results demonstrate that the effectiveness of ABARC remains robust regardless of these variations.

In smaller graph samples, ABARC demonstrates high effectiveness due to the limited number of nodes and edges, which facilitates easier manipulation and more apparent impact of the adversarial perturbations.

As the graph size increases, the challenge of embedding the backdoor while maintaining evasiveness also increases. However, ABARC has been designed to scale effectively, ensuring that the attack remains potent even in larger graph samples. This is achieved by building a dynamic trigger pattern and adaptively selecting nodes or node features as triggers based on the graph sample node scale and node feature dimensions, maximizing the impact with minimal changes.

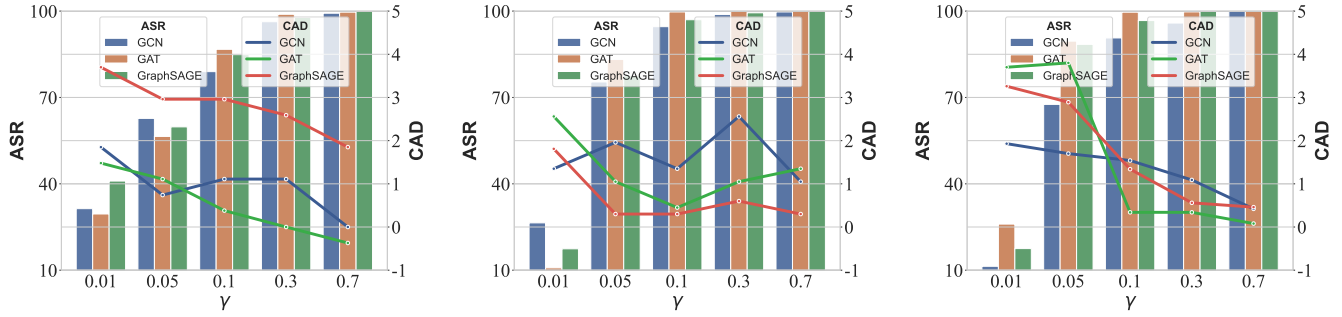
VII. CONCLUSION

In conclusion, this paper addresses the limitations of fixed pattern triggers and the lack of reasonable constraints of



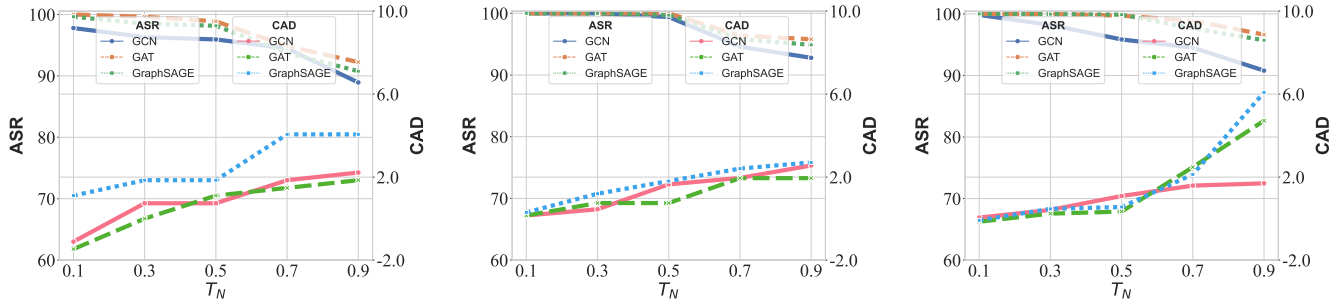
(a) Cora (b) CiteSeer (c) Flickr

Fig. 11. Detection results of ABARC and other attacks by NC on the node-level task using GCN.



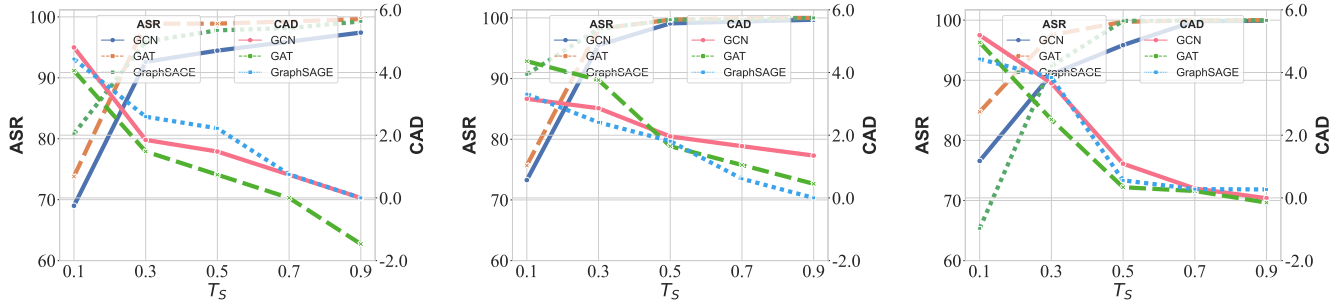
(a) Cora (b) CiteSeer (c) Flickr

Fig. 12. The attack performance of ABARC under various feature trigger proportion γ on node-level tasks when $T_N = 0.5$ and $T_S = 0.5$.



(a) Cora (b) CiteSeer (c) Flickr

Fig. 13. The attack performance of ABARC under various similarity threshold T_N on node-level tasks when $\gamma = 0.3$ and $T_S = 0.5$.



(a) Cora (b) CiteSeer (c) Flickr

Fig. 14. The attack performance of ABARC under various adaptive edge-pruning threshold T_S on node-level tasks when $\gamma = 0.3$ and $T_G = 0.5$.

backdoor attacks against GNNs for graph-level and node-level tasks. The proposed adaptive backdoor attacks, ABARC, effectively manipulate model behavior while maintaining evasiveness, highlighting the significance of security considerations in GNN applications.

A. Limitations

However, this paper still falls short in terms of a comprehensive semantic constraint design, resulting in a lack of evasiveness of the attacks. The constraints designed in this paper consider sample similarity, sample feature value range, and feature value type, ensuring the trigger's rationality to a certain extent. However, it is still a relatively broad constraint, lacking in-depth consideration of the meaning of graph sample nodes. For example, for molecular structure graph data, nodes represent atoms, and chemical bonds connect atoms. The valence electrons contained in the connected atomic locks should correspond. The dynamic trigger mode designed in this paper cannot achieve a high attack success rate under the premise of meeting this condition.

B. Defense Methods

ABARC can be defended from the following two aspects:

Defense can be carried out through **data filtering**. Since ABARC lacks comprehensive semantic constraints, defenders can filter graph data samples according to their characteristics before model training. However, this method relies on an in-depth understanding of graph data samples, and it is difficult to find a standard filtering method for various graph data. This complexity underscores the need for further research and the development of more effective filtering methods.

Furthermore, defense can also be carried out through **model comparison**. While ABARC ensures a high attack success rate, it minimizes the model's classification accuracy decline for normal data. However, the model's classification accuracy for normal data will still decline. Therefore, defenders can use their own reliable data sets to train a new model and determine whether the model has been attacked by comparing the classification accuracy of the new model and the suspicious model for normal data. Since the classification accuracy of the model attacked by ABARC for normal data has only slightly decreased, determining whether the suspicious model has been attacked remains a complex issue that requires further research. Simultaneously, better model comparison methods can also be further studied.

C. Future Research

There is a need for the development of more sophisticated and evasive dynamic trigger patterns, which could serve as a valuable avenue for future in-depth research. Furthermore, one of the future research directions could be the in-depth study of the working principle of GNNs to analyze and explain the security threats they face; at the same time, more effective and evasive backdoor attacks based on GNNs' inherent flaws could be developed. Another research direction could explore additional techniques to enhance the robustness of GNNs against backdoor attacks. Further investigation into advanced defense mechanisms and techniques to detect and mitigate backdoor attacks would be valuable. Moreover, exploring the impact of backdoor attacks on more complex graph structures, such as multi-relational graphs and heterogeneous graphs, would extend our understanding of their implications in real-world scenarios. Additionally, research into addressing backdoor attacks in federated learning settings, where GNN models are trained on distributed data sources, presents an interesting avenue for exploration. Overall, advancing research in GNN security will be crucial to safeguarding the integrity and reliability of graph-based applications in various domains. Besides, once ABARC is more thoroughly explored, its application or similar methodologies in other neural networks or against different types of attacks may be investigated.

REFERENCES

[1] C. Liu, L. Zhu, X. He, and J. Chen, "Enabling privacy-preserving shortest distance queries on encrypted graph data," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 192–204, 2021. [Online]. Available: https://doi.org/10.1109/TDSC.2018.2880981

[2] J. J. Irwin, T. Sterling, M. M. Mysinger, E. S. Bolstad, and R. G. Coleman, "Zinc: A free tool to discover chemistry for biology," *Journal of Chemical Information and Modeling*, vol. 52, no. 7, pp. 1757–1768, 2012. [Online]. Available: https://doi.org/10.1021/ci3001277

[3] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in Neural Information Processing Systems*, vol. 30, 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf

[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the 2016 International Conference on Learning Representations*, 2016. [Online]. Available: https://openreview.net/forum?id=SJU4ayYgl

[5] Y. Cao, H. Jiang, Y. Deng, J. Wu, P. Zhou, and W. Luo, "Detecting and mitigating ddos attacks in sdn using spatial-temporal graph convolutional network," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 6, pp. 3855–3872, 2022. [Online]. Available: https://doi.org/10.1109/TDSC.2021.3108782

[6] S. Wang and S. Y. Philip, "Heterogeneous graph matching networks: Application to unknown malware detection," in *Proceedings of the 2019 IEEE International Conference on Big Data*, 2019, pp. 5401–5408. [Online]. Available: https://doi.org/10.1109/BigData47090.2019.9006464

[7] H. Altae-Tran, B. Ramsundar, A. S. Pappu, and V. Pande, "Low data drug discovery with one-shot learning," *ACS Central Science*, vol. 3, no. 4, pp. 283–293, 2017. [Online]. Available: https://doi.org/10.1021/acscentsci.6b00367

[8] B. Wang, J. Jia, and N. Z. Gong, "Graph-based security and privacy analytics via collective classification with joint weight learning and propagation," in *Proceedings of the 2019 Network and Distributed System Security Symposium*, 2019. [Online]. Available: https://doi.org/10.14722/ndss.2019.23226

[9] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 974–983. [Online]. Available: https://doi.org/10.1145/3219819.3219890

[10] H. K. Ahmad, B. Liu, B. A. Muhammad, and M. Umar, "Prebige: Course recommendation using course prerequisite relation embedding and bipartite graph embedding," *Journal of Networking and Network Applications*, pp. 161–171, 2022. [Online]. Available: https://doi.org/10.33969/J-NaNA.2022.020404

[11] B. A. Muhammad, B. Liu, H. K. Ahmad, M. Umar, and K. Qin, "Gnn-ls: Alearning style prediction in online environments using graph neural networks," *Journal of Networking and Network Applications*, pp. 172–182, 2022. [Online]. Available: https://doi.org/10.33969/J-NaNA.2022.020405

[12] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 1115–1124. [Online]. Available: https://proceedings.mlr.press/v80/dai18b.html

[13] K. Xu, H. Chen, S. Liu, P. Y. Chen, T. W. Weng, M. Hong, and X. Lin, "Topology attack and defense for graph neural networks: An optimization perspective," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 3961–3967. [Online]. Available: https://www.ijcai.org/proceedings/2019/0550.pdf

[14] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2847–2856. [Online]. Available: https://doi.org/10.1145/3219819.3220078

[15] Y. Xiao, Z. Xing, A. X. Liu, L. Bai, Q. Pei, and L. Yao, "Cure-gnn: A robust curvature-enhanced graph neural network against adversarial attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 5, pp. 4214–4229, 2023. [Online]. Available: https://doi.org/10.1109/TDSC.2022.3211955

[16] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," arXiv preprint arXiv:1708.06733, 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1708.06733

[17] L. Zhang, P. Liu, Y.-H. Choi, and P. Chen, "Semantics-preserving reinforcement learning attack against graph neural networks for malware detection," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 2, pp. 1390–1402, 2023. [Online]. Available: https://doi.org/10.1109/TDSC.2022.3153844

[18] Z. Zhang, J. Jia, B. Wang, and N. Z. Gong, "Backdoor attacks to graph neural networks," in *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*, 2021, pp. 15–26. [Online]. Available: https://doi.org/10.1145/3450569.3463560

[19] J. Xu, M. Xue, and S. Picek, "Explainability-based backdoor attacks against graph neural networks," in *Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning*, 2021, pp. 31–36. [Online]. Available: https://doi.org/10.1145/3468218.3469046

[20] Z. Xi, R. Pang, S. Ji, and T. Wang, "Graph backdoor," in *Proceedings of the 30th USENIX Security Symposium*, 2021, pp. 1523–1540. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/xi

[21] E. Dai, M. Lin, X. Zhang, and S. Wang, "Unnoticeable backdoor attacks on graph neural networks," in *Proceedings of the 2023 ACM Web Conference*, 2023, pp. 2263–2273. [Online]. Available: https://doi.org/10.1145/3543507.3583392

[22] M. Li, A. Micheli, Y. G. Wang, S. Pan, P. Lió, G. S. Gnecco, and M. Sanguineti, "Guest editorial: Deep neural networks for graphs: Theory, models, algorithms, and applications," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 4, pp. 4367–4372, 2024. [Online]. Available: https://doi.org/10.1109/TNNLS.2024.3371592

[23] J. Li, R. Zheng, H. Feng, M. Li, and X. Zhuang, "Permutation equivariant graph framelets for heterophilous graph learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 9, pp. 11 634–11 648, 2024. [Online]. Available: https://doi.org/10.1109/TNNLS.2024.3370918

[24] M. Li, L. Zhang, L. Cui, L. Bai, Z. Li, and X. Wu, "Blog: Bootstrapped graph representation learning with local and global regularization for recommendation," *Pattern Recognition*, vol. 144, pp. 109 874:1–109 874:13, 2023. [Online]. Available: https://doi.org/10.1016/j.patcog.2023.109874

[25] C. Huang, M. Li, F. Cao, H. Fujita, Z. Li, and X. Wu, "Are graph convolutional networks with random weights feasible?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 2751–2768, 2023. [Online]. Available: https://doi.org/10.1109/TPAMI.2022.3183143

[26] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/hash/d80b7040b773199015de6d3b4293c8ff-Abstract.html

[27] Q. Huang, M. Yamada, Y. Tian, D. Singh, and Y. Chang, "Graphlime: Local interpretable model explanations for graph neural networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 7, pp. 6968–6972, 2022. [Online]. Available: https://doi.org/10.1109/TKDE.2022.3187455

[28] H. Zhang, J. Chen, L. Lin, J. Jia, and D. Wu, "Graph contrastive backdoor attacks," in *Proceedings of the 40th International Conference on Machine Learning*, 2023, pp. 40 888–40 910. [Online]. Available: https://proceedings.mlr.press/v202/zhang23e/zhang23e.pdf

[29] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, pp. 52–74, 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1709.05584

[30] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proceedings of the 2018 International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ

[31] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *Advances in Neural Information Processing Systems*, vol. 31, 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/hash/e77dbaf6759253c7c6d0efc5690369c7-Abstract.html

[32] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," in *Proceedings of the 2018 International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=SyJ7ClWCb

[33] Y. Ji, X. Zhang, S. Ji, X. Luo, and T. Wang, "Model-reuse attacks on deep learning systems," in *Proceedings of the 25th ACM Conference on Computer and Communications Security*, 2018, pp. 349–363. [Online]. Available: https://doi.org/10.1145/3243734.3243757

[34] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *Proceedings of the 25th Annual Network And Distributed System Security Symposium*, 2018. [Online]. Available: https://doi.org/10.14722/ndss.2018.23291

[35] X.-S. Li, X. Liu, L. Lu, X.-S. Hua, Y. Chi, and K. Xia, "Multiphysical graph neural network (mp-gnn) for covid-19 drug design," *Briefings in Bioinformatics*, vol. 23, no. 4, p. bbac231, 2022. [Online]. Available: https://doi.org/10.1093/bib/bbac231

[36] X. Shen, S. Zhang, J. Long, C. Chen, M. Wang, Z. Cui, B. Chen, and T. Tan, "A highly sensitive model based on graph neural networks for enzyme key catalytic residue prediction," *Journal of Chemical Information and Modeling*, vol. 63, no. 14, pp. 4277–4290, 2023. [Online]. Available: https://doi.org/10.1021/acs.jcim.3c00273

[37] F. Zhou, C. Cao, K. Zhang, G. Trajcevski, T. Zhong, and J. Geng, "Meta-gnn: On few-shot node classification in graph meta-learning," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 2357–2360. [Online]. Available: https://doi.org/10.1145/3357384.3358106

[38] Z. Tian, Y. Liu, J. Sun, Y. Jiang, and M. Zhu, "Exploiting group information for personalized recommendation with graph neural networks," *ACM Transactions on Information Systems*, vol. 40, no. 2, pp. 1–23, 2021. [Online]. Available: https://doi.org/10.1145/3464764

[39] K. Riesen and H. Bunke, "Iam graph database repository for graph based pattern recognition and machine learning," in *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshop, SSPR & SPR 2008, Orlando, USA, December 4-6, 2008. Proceedings*, 2008, pp. 287–297. [Online]. Available: https://doi.org/10.1007/978-3-540-89689-0_33

[40] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, A. J. Smola, and H. P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, pp. i47–56, 2005. [Online]. Available: https://doi.org/10.1093/bioinformatics/bti1007

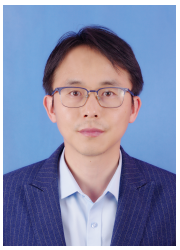
[41] M. Neuhaus and H. Bunke, "A graph matching based approach to fingerprint classification using directional variance," in *Proceedings of the 2005 International Conference on Audio-and Video-Based Biometric Person Authentication*, 2005, pp. 191–200. [Online]. Available: https://doi.org/10.1007/11527923_20

[42] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *Proceedings of the 33rd International Conference on Machine Learning*, 2016, pp. 40–48. [Online]. Available: https://proceedings.mlr.press/v48/yanga16.html

[43] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93–93, 2008. [Online]. Available: https://doi.org/10.1609/aimag.v29i3.2157

[44] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Accurate, efficient and scalable graph embedding," in *Proceedings of the 2019 IEEE International Parallel and Distributed Processing Symposium*, 2019. [Online]. Available: https://doi.org/10.1109/IPDPS.2019.00056

[45] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy*, 2019, pp. 707–723. [Online]. Available: https://doi.org/10.1109/SP.2019.00031

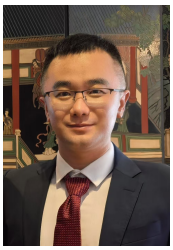


Xuewen Dong received his B.E., M.S. and Ph.D. degrees in Computer Science and Technology from the Xidian University, China, in 2003, 2006 and 2011, respectively. From 2016 to 2017, he was with the Oklahoma State University of USA as a visiting scholar. He is currently a professor in the School of Computer Science and Technology, Xidian University. His research interests include wireless network security and Blockchain.

Jiachen Li received his B.E. and B.S. degrees in Computer Science and Technology from the Xidian University, China, in 2021 and 2024, respectively. His research interests include graph neural networks and backdoor attacks.

Shujun Li (M'2008, SM'2012) received his B.E. degree in Information Science and Engineering and his Ph.D. degree in Information and Communication Engineering from Xi'an Jiaotong University, China, in 1997 and 2003, respectively. He is Professor of Cyber Security and Director of the Institute of Cyber Security for Society (iCSS), University of Kent, U.K. His research interests are mostly about inter-disciplinary topics related to cyber security and privacy, human factors, digital forensics and cybercrime, multimedia computing, AI and data science. His work covers multiple application domains, including but not limited to cybercrime, social media analytics, digital health, smart cities, smart homes, and e-tourism. He received multiple awards and honors including the 2022 IEEE Transactions on Circuits and Systems Guillemin-Cauer Best Paper Award.

Zhichao You received his B.E. degree in Information and Computing Science from South China Agricultural University, Guangzhou, China in 2018, and his M.S. degree in Computer Science and Technology from Xidian University, China in 2021. He is now pursuing his Ph.D. degree at Xidian University. His research interests include federated learning and wireless network security.



Qiang Qu received the PhD degree from Aarhus University, Aarhus, Denmark, supported by the GEOCrowd project under Marie Skodowska-Curie Actions. He is now a full professor with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. His current research interests include large-scale data management and mining, blockchain and data intelligence.

Yaroslav Kholodov had been a senior lecturer at the Department of Computational Mathematics of the Moscow Institute of Physics and Technology (MIPT) from 2021. In 2007, he became an Associate Professor. During 2009-2012 Yaroslav Kholodov had been a Chairman of the Organizing Board of the Annual MIPT School in High-Performance Computing for Young Scholars. During 2010-2012 Yaroslav had been the Head of the MIPT research and educational center "High-Performance Computing and Distributed Computer Systems". His key research area is represented by intelligent analysis of traffic data and road traffic modeling using adaptive control algorithms.

Yulong Shen received his B.S. and M.S. degrees in Computer Science and his Ph.D. degree in Cryptography from Xidian University, China, in 2002, 2005 and 2008, respectively. He is currently a Professor with the School of Computer Science and Technology, Xidian University, and also an Associate Director of the Shaanxi Key Laboratory of Network and System Security. He has also served on the technical program committees of several international conferences, including the NANA, the ICEBE, the INCoS, the CIS, and the SOWN. His research interests include wireless network security and cloud computing security.